**Pl/sql procedures and fucnctions**

**create or replace procedure Gen\_bill(rno in number)**

as

rec bill%rowtype;

a number;

cursor billdata is (select \*from bill where rollno=rno and dop between to\_char(sysdate-1/48,'dd-mon-yyyy hh:mi:ss pm') and to\_char(sysdate,'dd-mon-yyyy hh:mi:ss pm'));

cursor name\_of\_student is (select name from student where roll=rno);

sname varchar2(50);

total\_amount number;

begin

a:=1;

total\_amount:=0;

dbms\_output.put\_line(' Date-' || sysdate);

dbms\_output.put\_line(' ');

open name\_of\_student ;

fetch name\_of\_student into sname;

sys.dbms\_output.put\_line('Name-' || sname);

close name\_of\_student;

open billdata;

loop

fetch billdata into rec;

if(a=1) then

dbms\_output.put\_line('Rollno-' || rec.rollno);

a:=0;

SYS.dbms\_output.put\_line(' ');

end if;

exit when billdata%notfound;

dbms\_output.put\_line(rec.isbn\_barcode || ' ' || rec.name\_of\_item ||' '|| rec.price|| ' '|| rec.discount ||' '|| rec.quantity||' '|| rec.net\_price );

total\_amount:=total\_amount + rec.net\_price;

end loop;

close billdata;

SYS.dbms\_output.put\_line(' ');

SYS.dbms\_output.put\_line(' Total Amount-'|| total\_amount || ' Rs');

end;

**execute gen\_bill(101503006);**

**create or replace procedure gen\_requirement**

as

cursor records is select \*from new\_book where (limit-quantity) >=5; -- to generate the number of books required for different books.

begin

dbms\_output.put\_line('isbn title qty');

for item in records loop

dbms\_output.put\_line(item.isbn||' '||item.title||' '|| (item.limit- item.quantity)); -- genrate the requirement table.

end loop;

end;

**execute gen\_requirement;**

select \* from new\_book;

create or replace trigger trg\_bill\_tb

after insert on new\_book\_student

for each row

declare

titleb new\_book.title%type; -- title for the insertion into bill table.

priceb new\_book.price%type; -- price for the insertion into bill table.

discountb new\_book.discount%type; -- dicount for the insertion into bill table.

net\_priceb bill.net\_price%type; -- to calculate the net price for the corresponding quantity.

begin

select title,price,discount into titleb,priceb,discountb from new\_book where isbn=:new.isbn;

net\_priceb:= ((priceb\*(100-discountb))/100)\* (:new.quantity);

insert into bill values(:new.roll,:new.isbn,titleb,priceb,discountb,:new.quantity,sysdate,net\_priceb); -- inserting values into bill table.

end;

execute gen\_bill(101503015);

select \* from bill;

select \* from new\_book\_student;

**insert into new\_book\_student values('978-1-56619-909-4',101503015,3,sysdate);**

create or replace trigger book\_decrease

before insert on new\_book\_student

for each row

declare

qty\_checker new\_book.quantity%type; -- to get quantity of the book in new\_book table.

begin

select quantity into qty\_checker from new\_book where isbn=:new.isbn;

if qty\_checker<=:new.quantity then -- to check quantity is eqaual to zero.

raise\_application\_error(-20002,'Insuffient number of books ');

else -- if quantity is not equal to zero than update.

update new\_book set quantity=quantity-:new.quantity where isbn= :new.isbn;

end if;

dbms\_output.put\_line('book quantity successfully updated in database');

end;

create or replace trigger book\_increase

after insert on bs\_nbk

for each row

begin

update new\_book set quantity=quantity+:new.quantity where isbn= :new.isbn ;

dbms\_output.put\_line('book quantity successfully updated in database');

end;

Create or replace

trigger fk\_new\_book\_student

before insert

on new\_book\_student

for each row

declare

myisbn new\_book\_student.isbn%type;

begin

myisbn := :new.isbn;

select isbn into myisbn from new\_book where isbn=myisbn;

exception

when no\_data\_found then

raise\_application\_error(-20002,'foreign key voilated');

end;

* **(Check) Primary key trigger for new\_book**

Create or replace

trigger pk\_isbn\_new\_book

before insert

on new\_book

for each row

declare

myisbn new\_book.isbn%type;

begin

myisbn:=null;

select isbn into myisbn from new\_book where isbn=:new.isbn;

if myisbn is not null then

raise\_application\_error(-20000, 'Primary key voilatd ');

end if;

exception

when no\_data\_found then

null;

end;

select \* from new\_book;

insert into new\_book values('978-1-56619-909-4','a',5,400,5,25,10);

* **(check)foreign key of bs\_nbk(pan\_no)**

Create or replace

trigger fk\_new\_book\_PAN

before insert

on bs\_nbk

for each row

declare

mypan book\_supplier.pan\_no%type;

begin

mypan := :new.pan\_no;

select pan\_no into mypan from book\_supplier where pan\_no=mypan;

exception

when no\_data\_found then

raise\_application\_error(-20002,'foreign key voilated');

end;

select \* from bs\_nbk;

insert into bs\_nbk values('ABCDE1111d','978-1-56619-909-4',12,7,750,sysdate);

* **foreign key ( isbn) on new\_book\_student**

Create or replace

trigger fk\_new\_book\_isbn

before insert

on bs\_nbk

for each row

declare

myisbn new\_book.isbn%type;

begin

myisbn := :new.isbn;

select isbn into myisbn from new\_book where isbn=myisbn;

exception

when no\_data\_found then

raise\_application\_error(-20002,'foreign key voilated');

end;

select \* from bs\_nbk;

insert into bs\_nbk values('ABCDE1111d','978-1-56619-910-4',12,7,750,sysdate);

* **foreign key(roll) on new\_book\_student**

Create or replace

trigger fk\_new\_book\_roll

before insert

on new\_book\_student

for each row

declare

myroll new\_book\_student.roll%type;

begin

myroll := :new.roll;

select roll into myroll from student where roll=myroll;

exception

when no\_data\_found then

raise\_application\_error(-20002,'foreign key voilated');

end;

* **procedure for inserting new\_book\_student**

//Main procedure 4 triggers running at back end

**//1)fk\_roll 2)fk\_isbn 3)inserting bill 4)updating new\_book**

create or replace PROCEDURE add\_nbk(myisbn in new\_book\_student.isbn%type,myroll in new\_book\_student.roll%type,myquantity in new\_book\_student.quantity%type) as

BEGIN

insert into new\_book\_student values(myisbn,myroll,myquantity,sysdate);

dbms\_output.put\_line('Row Added Successfully');

END;

**execute add\_nbk('978-1-56619-917-4',101503006,3);**

* **RECORDS OF PREVIOUS 2 DAYS**

create or replace procedure check\_record(days in number)

as

rec bill%rowtype;

cursor records is (select \*from bill where dop between to\_char(sysdate-days,'dd-mon-yyyy hh:mi:ss pm') and to\_char(sysdate,'dd-mon-yyyy hh:mi:ss pm'));

total\_sale number;

begin

total\_sale:=0;

open records ;

loop

fetch records into rec;

exit when records%notfound;

total\_sale:=total\_sale+rec.net\_price;

dbms\_output.put\_line( rec.rollno || ' '||rec.isbn\_barcode || ' ' || rec.name\_of\_item ||' '|| rec.price|| ' '|| rec.discount ||' '|| rec.quantity||' '|| rec.net\_price );

end loop;

close records;

dbms\_output.put\_line(' ');

dbms\_output.put\_line('Total sale- '||total\_sale|| ' Rs');

end;

**execute check\_record(1/4);**

**execute check\_record(2);**

* **Updating**create or replace procedure update\_discount(myisbn in new\_book.isbn%type,mydiscount in new\_book.discount%type)

as

begin

update new\_book set discount =mydiscount where isbn= myisbn;

end;

execute update\_discount('978-1-56619-909-4',6);

* **Add new Supplier**

set serveroutput on;

declare

PROCEDURE add\_nb\_supplier(myname in book\_supplier.name%type,myaddress in book\_supplier.address%type,mypan\_no in book\_supplier.pan\_no%type ) IS

value\_to\_large exception;

unique\_cons\_voilated exception;

PRAGMA EXCEPTION\_INIT(value\_to\_large, -12899);

PRAGMA EXCEPTION\_INIT(unique\_cons\_voilated,-1);

BEGIN

insert into book\_supplier values(myname,myaddress,mypan\_no);

Exception

when value\_to\_large then

dbms\_output.put\_line('Value to large than specified column width');

when unique\_cons\_voilated then

dbms\_output.put\_line('unique cons voilated');

END;

begin

add\_nb\_supplier('gupta','Lahori gate,Amritsar','ABCDE1117F');

dbms\_output.put\_line('Row Added Successfully');

end;

* **Add stat supplier**

set serveroutput on;

declare

PROCEDURE add\_stat\_supplier(mypan\_no in book\_supplier.pan\_no%type,myname in book\_supplier.name%type,myaddress in book\_supplier.address%type ) IS

value\_to\_large exception;

unique\_cons\_voilated exception;

PRAGMA EXCEPTION\_INIT(value\_to\_large, -12899);

PRAGMA EXCEPTION\_INIT(unique\_cons\_voilated,-1);

BEGIN

insert into stat\_supplier values(mypan\_no,myname,myaddress);

dbms\_output.put\_line('Row Added Successfully');

Exception

when value\_to\_large then

dbms\_output.put\_line('Value to large than specified column width');

when unique\_cons\_voilated then

dbms\_output.put\_line('unique cons voilated');

END;

begin

add\_stat\_supplier('ABCDE1121F','Sangal Ram','Chherata amritsar');

end;

* **Add new Student**

set serveroutput on;

declare

PROCEDURE add\_student(myroll in student.roll%type,myname in student.name%type ) IS

value\_to\_large exception;

unique\_cons\_voilated exception;

PRAGMA EXCEPTION\_INIT(value\_to\_large, -12899);

PRAGMA EXCEPTION\_INIT(unique\_cons\_voilated,-1);

BEGIN

insert into student values(myroll,myname);

dbms\_output.put\_line('Row Added Successfully');

Exception

when value\_to\_large then

dbms\_output.put\_line('Value to large than specified column width');

when unique\_cons\_voilated then

dbms\_output.put\_line('unique cons voilated');

END;

begin

add\_student(101503030,'Ankit shubham');

end;

* **Add New Book via procedure**

set serveroutput on;

declare

PROCEDURE add\_new\_book(myisbn in new\_book.isbn%type,mytitle in new\_book.title%type,myedition in new\_book.edition%type,myprice in new\_book.price%type ) IS

value\_to\_large exception;

unique\_cons\_voilated exception;

PRAGMA EXCEPTION\_INIT(value\_to\_large, -12899);

PRAGMA EXCEPTION\_INIT(unique\_cons\_voilated,-1);

BEGIN

insert into new\_book values(myisbn,mytitle,myedition,myprice);

dbms\_output.put\_line('Row Added Successfully');

Exception

when value\_to\_large then

dbms\_output.put\_line('Value to large than specified column width');

when unique\_cons\_voilated then

dbms\_output.put\_line('unique cons voilated');

END;

begin

add\_new\_book('978-1-56619-917-4','Manorma year book',1,800);

end;

* **Add old book procedure**

set serveroutput on;

declare

PROCEDURE add\_old\_book(myisbn in old\_book.isbn%type,mytitle in old\_book.title%type,myedition in old\_book.edition%type,myprice in old\_book.price%type,myquantity in old\_book.quantity%type ) IS

value\_to\_large exception;

unique\_cons\_voilated exception;

PRAGMA EXCEPTION\_INIT(value\_to\_large, -12899);

PRAGMA EXCEPTION\_INIT(unique\_cons\_voilated,-1);

BEGIN

insert into old\_book values(myisbn,mytitle,myedition,myprice,myquantity);

dbms\_output.put\_line('Row Added Successfully');

Exception

when value\_to\_large then

dbms\_output.put\_line('Value to large than specified column width');

when unique\_cons\_voilated then

dbms\_output.put\_line('unique cons voilated');

END;

begin

add\_old\_book('978-1-56619-917-4','Manorma year book',1,800,5);

end;

* **add stationery**

declare

PROCEDURE add\_stationery(mybarcode in stationery.barcode%type,mytitle in stationery.company%type,myname\_of\_item in stationery.name\_of\_item%type,myprice in stationery.price%type) IS

value\_to\_large exception;

unique\_cons\_voilated exception;

PRAGMA EXCEPTION\_INIT(value\_to\_large, -12899);

PRAGMA EXCEPTION\_INIT(unique\_cons\_voilated,-1);

BEGIN

insert into stationery values(mybarcode,mytitle,myname\_of\_item,myprice);

dbms\_output.put\_line('Row Added Successfully');

Exception

when value\_to\_large then

dbms\_output.put\_line('Value to large than specified column width');

when unique\_cons\_voilated then

dbms\_output.put\_line('unique cons voilated');

END;

begin

add\_stationery('111111111116','NonDust','Eraser',8);

end;

* **Display book\_Suppliers**

set serveroutput on;

declare

rec book\_supplier%rowtype;

cursor mc is

select \* from book\_supplier;

PROCEDURE show\_stat\_supplier IS

BEGIN

open mc;

dbms\_output.put\_line('NAME'||' '||'ADDRESS'||' '||'PAN\_NO');

loop

fetch mc into rec;

EXIT WHEN mc%NOTFOUND;

dbms\_output.put\_line(rec.name ||' '|| rec.address ||' '|| rec.pan\_no);

end loop;

dbms\_output.put\_line('Number of Recors fetched -->'||mc%rowcount);

close mc;

END;

begin

show\_stat\_supplier();

end;

* **Show Stationery Supplier**

set serveroutput on;

declare

rec stat\_supplier%rowtype;

cursor mc is

select \* from stat\_supplier;

PROCEDURE show\_stat\_supplier IS

BEGIN

open mc;

dbms\_output.put\_line('Pan no '||' '||'Name'||' '||'Address');

loop

fetch mc into rec;

EXIT WHEN mc%NOTFOUND;

dbms\_output.put\_line(rec.pan\_no ||' '|| rec.name ||' '|| rec.address);

end loop;

dbms\_output.put\_line('Number of Recors fetched -->'||mc%rowcount);

close mc;

END;

begin

show\_stat\_supplier();

end;

* **Show new Book**

set serveroutput on;

declare

rec new\_book%rowtype;

cursor mc is

select \* from new\_book;

PROCEDURE show\_new\_book IS

BEGIN

open mc;

dbms\_output.put\_line('ISBN '||' '||'TITLE'||' '||'EDITION' ||' '||'PRICE');

loop

fetch mc into rec;

EXIT WHEN mc%NOTFOUND;

dbms\_output.put\_line(rec.isbn ||' '|| rec.title ||' '|| rec.edition ||' '||rec.price);

end loop;

dbms\_output.put\_line('Number of Recors fetched -->'||mc%rowcount);

close mc;

END;

begin

show\_new\_book();

end;

* **Show Old book**

set serveroutput on;

declare

rec old\_book%rowtype;

cursor mc is

select \* from old\_book;

PROCEDURE show\_old\_book IS

BEGIN

open mc;

dbms\_output.put\_line('ISBN '||' '||'TITLE'||' '||'EDITION' ||' '||'PRICE'||' QUANTITY');

loop

fetch mc into rec;

EXIT WHEN mc%NOTFOUND;

dbms\_output.put\_line(rec.isbn ||' '|| rec.title ||' '|| rec.edition ||' '||rec.price||' '||rec.quantity);

end loop;

dbms\_output.put\_line('Number of Recors fetched -->'||mc%rowcount);

close mc;

END;

begin

show\_old\_book();

end;

* **show stationery**

set serveroutput on;

declare

rec stationery%rowtype;

cursor mc is

select \* from stationery;

PROCEDURE show\_stationery IS

BEGIN

open mc;

dbms\_output.put\_line('BARCODE '||' '||'COMPANY'||' '||'EDITION');

loop

fetch mc into rec;

EXIT WHEN mc%NOTFOUND;

dbms\_output.put\_line(rec.barcode ||' '|| rec.company ||' '|| rec.name\_of\_item ||' '||rec.price);

end loop;

dbms\_output.put\_line('Number of Recors fetched -->'||mc%rowcount);

close mc;

END;

begin

show\_stationery();

end;

* **show student**

set serveroutput on;

declare

rec student%rowtype;

cursor mc is

select \* from student;

PROCEDURE show\_student IS

BEGIN

open mc;

dbms\_output.put\_line('ROLL '||' '||'NAME');

loop

fetch mc into rec;

EXIT WHEN mc%NOTFOUND;

dbms\_output.put\_line(rec.roll ||' '|| rec.name );

end loop;

dbms\_output.put\_line('Number of Recors fetched -->'||mc%rowcount);

close mc;

END;

begin

show\_student();

end;

* **insert new\_book\_student(New Book Sold)**

declare

begin

add\_nbk('978-1-56619-917-4',101503006,6,3);

end;

Trigger Foreign key for isbn in new\_book\_student(fk for new\_book\_student) without using pragma exception

Create or replace

trigger fk\_new\_book\_student

before insert

on new\_book\_student

for each row

declare

myisbn new\_book\_student.isbn%type;

begin

myisbn := :new.isbn;

select isbn into myisbn from new\_book where isbn=myisbn;

exception

when no\_data\_found then

raise\_application\_error(-20002,'foreign key voilated');

end;

Trigger Foreign key for roll in new\_book\_student(fk for new\_book\_student) without using pragma exception

Create or replace

trigger fk\_new\_book\_roll

before insert

on new\_book\_student

for each row

declare

myroll new\_book\_student.roll%type;

begin

myroll := :new.roll;

select roll into myroll from student where roll=myroll;

exception

when no\_data\_found then

raise\_application\_error(-20002,'foreign key voilated');

end;